AD-A050 802 SACRAMENTO AIR LOGISTICS CENTER MCCLELLAN AFB CA DATA--ETC F/G 5/1
SOFTWARE ENGINEERING PROJECT MANAGEMENT: A SURVEY CONCERNING U.--ETC(U)
NOV 77 R.H. THAYER, J. H. LEHMAN
UNCLASSIFIED SM-ALC/ACD-TR-77-02 N/L

1 OF 1
AD-
A050802

END
DATE
FILMED
2-80
DDC

AD A050802

AD NO.
DDC FILE COPY

SM-ALC/ACD-TR-77-Ø2

1 November 1977

SOFTWARE ENGINEERING PROJECT MANAGEMENT: A SURVEY CONCERNING U.S. AEROSPACE INDUSTRY MANAGEMENT OF SOFTWARE DEVELOPMENT PROJECTS

(Interim Report)

Richard H. Thayer
Sacramento Air Logistics Center
Air Force Logistics Command
McClellan AFB, CA 95652

and

John H. Lehman
California State University
Sacramento, CA 95819

Approved for Public Release

Unlimited Distribution

DEPARTMENT OF THE AIR FORCE
HEADQUARTERS SACRAMENTO AIR LOGISTICS CENTER (AFLC)
McCLELLAN AIR FORCE BASE, CALIFORNIA 95652

REPRODUCED BY
NATIONAL TECHNICAL INFORMATION SERVICE
U.S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SM-ALC/ACD TR-77-02 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Software Engineering Project Management: A Survey Concerning U.S. Aerospace Industry Management of Software Development Projects. (Interim Report)		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER SM-ALC/ACD TR-77-02
7. AUTHOR(s) Richard H. Thayer and John H. Lehman (California State University, Sacramento)		8. CONTRACT OR GRANT NUMBER(s) Not Applicable
9. PERFORMING ORGANIZATION NAME AND ADDRESS Data Automation Branch Sacramento Air Logistics Center/ACD ✓ McClellan Air Force Base, California 95652		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Not Applicable
11. CONTROLLING OFFICE NAME AND ADDRESS Same as Nr 9		12. REPORT DATE 1 November 1977
		13. NUMBER OF PAGES 19
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same as Nr 9		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release
Unlimited Distribution

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for Public Release
Unlimited Distribution

18. SUPPLEMENTARY NOTES

Prepared in cooperation with The American Institute of Aeronautics and Astronautics (AIAA) Technical Committee on Computer Systems.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Engineering Project Management, Software Development, Survey Project Management.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Project management is clearly a part of software engineering, and its effective employment plays a major role in reducing the problems associated with delivering software within estimated time and cost. The question this paper addresses is: "What is the state-of-the-art in software engineering project management today?" In an attempt to provide this answer, a survey of highly qualified executive managers of major U.S. aerospace corporations was conducted. This paper reports on that survey describing how these corporations manage software development projects, discusses the major differences in methods used, and how software engineering project management might be improved.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

 SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

NOTICE

THIS DOCUMENT HAS BEEN REPRODUCED FROM THE BEST COPY FURNISHED US BY THE SPONSORING AGENCY. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE.

15 December 1977

PREFACE

This paper reports the partial results of a survey conducted during the spring and summer of 1977 on how the U.S. aerospace industry manages its software engineering projects. The survey was conducted on a sample set of the U.S. aerospace industry as represented by membership in the American Institute of Aeronautics and Astronautics (AIAA) Technical Committee on Computer Systems. The impetus behind this survey was to collect data for analysis in the preparation of paper on software engineering project management to be presented at the AIAA Conference, Computers in Aerospace, 31 October through 2 November 1977, in Los Angeles, California. This report, which can also be found in the conference proceedings, A Collection of Technical Papers, under the title "Software Engineering Project Management: A State-of-the-Art Report" represents only a portion of the data.

Time constraints dictated that the paper be at the printers by 23 September 1977. On 6 September 1977 decision was made "to go with" the data on hand, which represented 29 surveys and 18 companies. By the time the actual presentation was given on 1 November 1977, fifty-two projects were reported, representing 33 companies, or 70% returned.

It is anticipated this report will be rewritten using the balance of the data received from the aerospace corporations that participated in the survey. Not only the data that was available on 1 November, which was used in the presentation itself at the conference, but other data that is still coming in as of this date.

The title of this report was changed on the cover sheet to more accurately reflect the contents, and to draw attention to the fact that this is an interim report. It is also anticipated that because of the voluminous amount of the data received in the survey, other reports will be made, not only by the original authors, but by other members of the AIAA Technical Committee on Computer Systems.

Richard H. Thayer, USAF

SOFTWARE ENGINEERING PROJECT MANAGEMENT: A STATE-OF-THE-ART REPORT

Richard H. Thayer*
Sacramento Air Logistics Center
Air Force Logistics Command
McClellan AFB, CA 95652

and

John H. Lehman
California State University
Sacramento, CA 95819

Abstract

Project management is clearly a part of software engineering, and its effective employment plays a major role in reducing the problems associated with delivering software (responsive to the users needs) within estimated time and cost. Software engineering project management must provide the necessary planning, organization, staffing, direction, and control. The question this paper addresses is: "What is the state-of-the-art in software engineering project management today?". In an attempt to provide this answer, a survey of major U.S. aerospace corporations was made. In response, highly qualified individuals reported on how they, their company, or department managed large software engineering projects. This paper includes survey results describing how these corporations manage software development projects, discussion and comments on the major differences in the methods used, and opinions on how software engineering project management might be improved.

Introduction

Software Engineering.

The failure of many software development projects to be delivered within schedule and cost, responsive to the users needs and to exhibit a high degree of maintainability and reliability is well documented and a major topic of discussion when any group of programmers gets together.

Software engineering was introduced in the 1970's to apply the stronger disciplines of engineering (in contrast to the "Art of Computer Programming") to the design of computer software in an attempt to solve the problem of, or at the very least reduce the severity of, software development failures. As a result of this approach, improvements have been made in software development techniques which have directly aided the programmer.

Similar advances have not been heralded in the area of software engineering project management, and only a few new techniques unique to this task have been developed.

Increased emphasis on the management aspects, as far as the authors can determine, dates from the NATO Conference on Software Engineering in 1968, and its successor, the 1969 NATO Conference on Software Engineering Techniques from which the following quotation was obtained.

> **In almost no department of computer science was it required that every student take a course in operations research and management science and yet I claim that to be a fundamental course in software engineering; as fundamental as automata theory and more fundamental than any mathematics course I can think of. [Perlis, Report NATO Conference on Software Engineering Techniques, 1969]**

Since that time the procedures generally grouped under the heading of "modern programmer productivity techniques" have had a wide, though not necessarily even, implementation by software development activities. Have management techniques changed to keep pace, or do the old procedures still apply? And, just what is the trend and the forms in software engineering project management today? These are the major questions we have attempted to address.

Data for this paper was collected from the members of the AIAA Technical Committee on Computer Systems who volunteered to participate in a project management survey. Our purpose is to provide a state-of-the-art report on how software engineering projects are managed in the firms of the aerospace industry in today's environment. The paper includes: 1) survey results in condensed form, 2) a discussion and comments on the major differences in the methods used, and, 3) an opinion on how software engineering project management might be improved.

Approach

The approach taken to gather data was to undertake a descriptive survey. A determination was made that the members of the American Institute of Aeronautics and Astronautics (AIAA) Technical Committee on Computer Systems would provide an excellent source and sample. Committee members represented 47 major corporations, or major corporate sub-divisions, and occupied top positions (e.g., Manager, Software and Computer Systems Engineering; Manager, Advanced Computer Technologies; Manager, Scientific Programming Department) within the firms. There can be no doubt that the participants were in an ideal position to report how the U.S. aerospace industry managed its software development projects.

Theoretical Model of a Software Engineering Project

It was the intent of the authors to

* Associate Fellow AIAA

identify and probe every aspect of software engineering project management. After rather lengthy biographical research it was concluded that no single reference, or group of references, was broad enough to cover the entire subject. Therefore, to assist in designing a questionnaire that was "complete" the authors built a theoretical model utilizing management processes, non-management processes, activities and products as elements. This model served as the basis from which a full range of questions, with their associated possible responses, was developed.

The Survey

The survey was conducted through a rather lengthy written questionnaire containing 229 numbered questions. But, beyond that, by using "packing techniques" approximately 925 separate responses were possible. The survey, running to 72 pages, was divided into three parts. Part One dealt with defining the total organization, management structure, requirements, and philosophy of the firm. It was intended to be answered by top management and provide the backdrop against which the individual projects would be viewed.

Part Two concerned questions about individual projects aimed at, and intended to be completed by, the project manager. It was also assumed that the project being reported on was or was almost completed. This part, containing the bulk of the questions, was divided into 8 sections. Segmentation was accomplished by grouping the questions under the functional headings of planning, organizing, staffing, directing and controling, and the technical categories of project identification, requirements specification and project results.

Part Three, consisting of general questions not project specific, but calling for evaluations, opinions and suggestions, was also intended to be completed by a project manager.

The questionnaire was designed to make analysis as easy as possible so responses were primarily of the multiple choice or fixed alternative types rather than open-ended or narrative in form. However, where the authors felt the possible number of alternatives was very large, or judgements were called for, open-ended or narrative type questions were used.

Method of Conducting Survey.

The survey was designed, written, tested, and implemented in the spring and summer of 1977. Initial contact was made in May of 1977 to determine which members of the committee would be interested, willing, and/or able to participate. Forty-five members, representing 35 companies, agreed to respond. An initial draft was completed in June 1977, and critiqued by approximately 25% of the total committee membership. The returns from this critique, along with other corrections, were incorporated in the final survey completed by the members in early September 1977.

Accuracy of the Survey.

Of the three basic methods in which this data could have been derived, (observation, document/record review, and survey) the survey afforded the only practical approach. Though observation and document/review would have been more accurate and perhaps provided greater uniformity, the financial and time constraints alone rendered this approach impractical. The survey, our only possible alternative, carried with it several possible infirmities. Good questionnaires are difficult to construct; it has been estimated that it takes eight hours to write one good question with its accompanying selection of responses. And, in spite of the authors best intentions, questions will not always mean the same thing to all people. Surveys are also difficult to administer. "Administer" is a euphemism for "get back on time".

Our questionnaire was probably no better or worse than other surveys in this respect. It was developed by two individuals over a period of three months and reviewed by perhaps a dozen experts in the data processing field prior to publication. Even before the returns started coming in, some problems of ambiguity were identified; however, based on the sheer magnitude of the survey and the ability to compare answers within each return, most of these ambiguities could be resolved.

Often, when respondents chose "other" and answered a question in a narrative form it was possible to place the response in a predefined category, or establish a new category in which other "other" responses could also be accumulated.

In instances where the answer to a preceding question dictated the response to following questions that were left blank, the answer was provided to assure tracking. An example of this might be a series of questions concerning HIPO diagrams in which only the first question was responded to with "Did not use HIPO diagrams".

The above procedures were applied in the absolute minimum number of cases, and we are certain that they did not distort the respondents intentions. In some few instances where uncertainty still existed, we either eliminated the response or confirmed the interpretation over the phone.

We believe that, though some inaccuracies exist due to interpretation, in aggregate the study meets its objective and does reflect the state-of-the-art in the management of software development projects as practiced in the aerospace industry today.

Survey Results

Participation.

In keeping with the highest traditions of software development there were both slippages and overruns. Each milestone was slipped by one week and no amount of extra effort or midnight oil could alter this perverse application of Parkinsons Law in which work expanded, with amazing regularity, to one week beyond the time allotted for its completion.

Of the 45 companies surveyed, slightly less than one-half completed the forms and returned them in time for our standard week-late cut off date. This allowed something less than two weeks to write and prepare the final paper so the decision was made to terminate any attempt to include data from late returns in this paper, but rather to update it for presentation at the conference in the fall. In some instances, where we have yielded to temptation and peeked ahead, we have found that no significant differences exist between the first 29 projects reported on and the subsequent arrivals.

Since each firm was contacted personally, both before and during survey time, it was anticipated that a very large majority of forms would ultimately be returned. It was not a lack of interest on the part of the participants, but, rather the short time available, the complexity of the questionnaire, and the prevalence of late summer vacations in the upper reaches of industry that have contrived to keep the initial response at just under 50%.

Each participant was given the option of: complete anonymity through full dissemination of their returned survey. In virtually every instance anonymity was opted for, so no reference to specific firms will be made in the body of the report and only summary/statistical data will be maintained or made available to interested parties.

The Firm.

Of the initial returns, approximately 75% of the firms were engaged primarily in manufacturing and 15% in engineering and technical support services. There was also one software house and one government agency. Average income for the firms ranged from "Between One and Ten Million Dollars" (one response) to, "In Excess of One Billion Dollars" (two responses). A rough approximation of mean income for all the companies included in this report is 290 million dollars annually, no mean sum. Only a small percentage of this income was derived from software development, however in most instances being less than 10%.

From the standpoint of employees, the number ranged from a low of 210 to a high of 50,000, while employees engaged in software development activities showed a low of 20 and a high of 1200 individuals, with 177 being the average.

Several questions address contract types: What types were used? What types were preferred? In analyzing this no specific "winner" emerged, and it is recognized that many factors go into the choice. Some contract types were clearly held in lower esteem than others. In general, cost plus types were most preferred when the company was in the position of contractor and firm fixed price the preference when the roles were reversed.

What emerged as the most preferred form of contract was the two-phased instrument in which phase one analyzes requirements, determines feasibility, and estimates cost. Phase two directs development. Only one firm discouraged their use, but only when they were contracting for development. Presumably it was felt that the expertise in-house was equal to the task of providing phase one analysis.

Only one firm payed incentives for early or on time project completion; and, whereas only one firm reported utilizing procedures in which programmers or analysts bid on specific tasks within development projects, two reported great to moderate success with this technique.

It was found that in aggregate, 94% of the development team members were permanent employees, with temporary hires and consultants accounting about equally for the remaining 6%. Though impossible to come up with an accurate percentage figure or ratio, the responses indicate clearly that the straight applications analyst or straight computer programmer is in a decided minority (probably less than 25%) with analysts/programmers being the rule.

Only one organization, the smallest with 20 individuals engaged in software development, reported not using any form of manual periodic progress reporting. Weekly status reports were the most common - 77%, project status, next - 61%, and significant change, third, - 46%.

Forty-six percent of the firms also reported using some form of automated project management system. Surprisingly, the second and third firms in size from a development staff viewpoint (averaging 650 people) did not use an automated project management system, and only slightly over 20% of those responding used system software to monitor development. As far as could be determined from a company standpoint, productivity indexes were in total disuse, though it was found later on that project managers did on occasion employ these techniques. Approximately 70% of the firms responding indicated that they had employed one or more of the procedures generally placed under the heading of "modern programmer productivity techniques", though only 40% appear to have made a substantial commitment. Of the

seven techniques listed (team concept, development support library, HIPOs, pseudo code, walk-thru's, top-down design, and top-down implementation), the employment of HIPOs has proven least accepted with only two firms reporting their use, one at the 10% level and growing, the other at 2% and in a decline.

Cost per line of code is a mandatory data processing survey question, so of course it was included. Costs varied from a low of $5.00 to a high of $330.00 - with an average at $78.78. It is, of course, a meaningless statistic given like this, but it is easy to remember.

Project Identification.

In collecting the data on specific projects we felt it important to obtain responses from those individuals most familiar with the development effort - the project manager or a senior member of his staff. In keeping with our requests that these "front line" supervisors provide the inputs, 78% of the respondents were indeed project managers with the remaining 22% occupying supervisory, staff, or technical positions just one step removed.

As for the projects, Table 1 provides a summary of the application/functional areas into which they fell. Many projects covered more than one application/functional area prompting us to come up with an additional category, "Multi-functional command and control systems." In this category we placed those projects that the respondent indicated covered many areas centering around a military command and control application.

Table 1

Application/Functional Areas

Application/Functional Area	Pct Reported
Commercial/business	7
Data acquisition/retrieval	3
Scientific/data reduction	14
Process control/embedded computers	28
Command and control systems	21
Management information systems	3
Communication systems	3
Computer systems software	3
Multi-functional command and control systems	17

About two-thirds of these projects involved totally new software development while the remaining one-third was either the continuation of a previously completed software system, a major modification of an existing capability, or both. About 54% of the projects used commercial off-the-shelf computer hardware, 42% employed special purpose computers, and one project used a combination commercial/special purpose system.

As would be expected in dealing with the U.S. aerospace industry, 78% of the projects were contracted for by the Federal Government, the balance being for other companies or in-house use. Table 2 lists the contract types used on the projects for which this data was reported.

Table 2

Contract Type Used

Contract Type	Pct Used
Firm fixed price	18
Cost	12
Cost sharing	4
Cost plus incentive fee	23
Cost plus award fee	8
Cost plus fixed fee	27
Time and materials	4
Basic ordering agreement	4

These contracts involved large sums of money ranging from $200,000 to half a billion dollars, with the total cost of all projects surveyed approximating one billion dollars. Software development accounted for $301,000,000 of this amount, distributed between $30,000 and $159,000,000.

With few exceptions, all of these projects were started subsequent to 1972, with 33% being completed in the 1975/1977 time frame. Fifty-two percent of the projects are still unfinished, but the majority of these are nearing completion. The languages most used (by a wide margin) were FORTRAN and some assembly language. Executable lines of code ranged from 5,000 to 1,200,000. Total executable lines of code reported on all projects were 4,073,800.

Requirement Specifications.

Had we been searching for someone who had been provided a perfect set of requirement specifications we would have come away sorely disappointed. As a general rule, the larger the system, the more

detailed the specifications. Yet, the more detailed the specification the greater the requirement for rewrite prior to proceeding with design. We asked the question: "On a scale of 1 to 7, with 1 being little more than the name of the system, and 7 being complete specifications down through preliminary design, how detailed were the specifications provided?"

Combining ratings and matching them with the project costs and specification rewrite requirements, we came up with the information in Table 3.

Table 3

Original Specification Detail Vs Percent Rewrite

Level of Specification Detail	Average System Cost	Pct Spec Rewrite Prior to Design Avg	High	Low	Nr Resp
1, 2 & 3	640,000	21	50	0	8
4 & 5	2,840,000	27	60	0	8
6 & 7	27,657,000	35	100	0	9

Of the specifications prepared in-house by the project managers' own organization, an average 23% rewrite was required whereas a 47% rewrite was called for when the user provided the specification. The customer or customer affiliate prepared the initial specifications in about 30% of the cases reported, the organization in 60%, and consultants or similar third party in the remaining 10%.

The reasons for specification rewrites were about equally divided between:

"Original requirements ambiguous and incomplete", and, "Continued Government redirection and problem rescoping".

One manager, however, considered specifications rewriting an immutable rule, a "normal and expected iteration of total system design". Table 4 lists the methods and techniques reported used in specification preparation.

It was interesting to note that no apparent relationship existed between successful projects and who originated the specifications or the degree of rewrite required.

Documentation. One of the major areas of customer concern was the type of documentation to be provided as part of the system. There was no discernible relationship between the number of document types required and system size. It should be noted that though the United States Government was the customer in the majority of cases, the number of document types specified as deliverable end items did not differ in any significant way between Governmental or non-Governmental customers. Three contracts called for 3 or fewer documents while all the rest required between 5 and 13. See Table 5 for a list of software documentation required by surveyed projects (28 projects reporting).

Table 4

Specification Preparation Techniques

Technique	Pct Used
Top-down (Hierarchy of function)	32
Structured flow	4
Formal requirements language	0
Phases where design and coding started before specifications were complete	32
In a manner to facilitate tracking software development from requirements through coding	44
In precise measureable terms to aid in development of acceptance tests	12

Table 5

Software Documentation Required by Customers

Document	Pct Required
Object listing	75
Source listing	93
Functional Description	82
Data Requirements Document	50
System/Subsystem Specifications	79
Program Specifications	71
Data Base Specifications	50
Users Manual	75
Computer Operations Manual	64
Program Maintenance Manual	32
Test Implementation Plan	75
Test Analysis Report	61
Other	29

Specific Customer Requirements. In addition to document types a host of other customer-defined requirements were also specified. These included standards, tech-

niques, limitations, and constraints, and are summarized in Table 6 (25 projects reporting).

Table 6

Specific Customer Requirement

Requirement	Pct Required
Specific computer	40
Adherence to storage limitations	36
Adherence to speed constraints	52
Adherence to specific language	48
Customer participation in design function	44
Customer participation in coding function	20
Adherence to prioritized requirements	28
Development under life-cycle-costing	8
Development under design-to-cost	8
Development under programming techniques	24
Portability	4
Adherence to human engineering techniques	36
Data system security	20
MIL-STD 5279 (or modified)	8
Specified types of review	68
Specified frequency of reviews	52
Special input data	44
Special output requirements	48
Test plan/procedures	20
A reliability figure	12
A maintainability goal	0
A warrantee of the software	4
A followup maintenance contract	24
Customer training	32

Planning.

Of the total time available to the project manager and his staff, approximately 11% was employed in planning and replanning. In most cases the project manager was "brought on board" shortly after project inception and participated actively in the planning activities. Table 7 breaks down the total planning time into its constituent parts (18 projects reporting).

In 19% of the projects reported the firm provided the manager with a formal planning guide. In 43% of the projects he was assisted in planning activities by the customer. Though two small projects reported producing no planning documents at all, the range and number of plans developed by the remainder is impressive, as can be seen in the Table 8.

Table 7

Allocation of Planning Time

Planning Function	Pct Time Spent
Developing an overall project management plan	22
Developing control procedures	21
Staff planning	17
Organizational planning	16
Quality assurance planning	12
Administrative planning	9
Other	3

In planning the development activities a number of recognized tools and procedures were employed. In two-thirds of the cases, projects were divided into phases to facilitate planning, and 62% of the projects used a work breakdown structure, generally shredded out to 5 levels.

Table 8

Planning Documents Prepared

Type of Plan	Pct Reported
Software development	85
Test	78
Training	67
Organization	70
Change control	70
Staffing	63
Review and reporting	59
Resource requirements	50
Documentation	56
Project management	48
Phase and/or delivery	37
Implementation	30
Data conversion	4
Other	7
None	8

Workload charts proved to be the most popular tool (used on 57% of the projects) followed by GANTT charts and some version of PERT (roughly 35% each).

Cost estimation, as everyone has suspected for some time, is far from a science. Table 9 shows the methods employed and the percentage of projects on which they were used. Some projects combined methods. For those clamoring for their own crystal ball (fourth method), it should be noted that the projects relying on that technique were each 6 months late and experienced significant cost overruns.

There is every indication that the development of quality assurance standards was also a major activity engaged in by project personnel. In interpreting Table 10 (16 projects reporting) it should be noted that in only two instances were both a company standard and a project unique standard applied at the same time. In every other case if the company had a standard it was used, and no project unique standard was developed (or maybe the other way around).

Table 9

Methods of Estimating Cost/Schedules

Method	Pct Used
Estimates based on a similar project	67
Formula	44
Cost and schedule dictated	15
Crystal ball (or similar means)	11
Provided by someone who has a knack for estimating correctly	4
Simulation techniques	0
Other	11

Only 20% of the participants reported not using a quality assurance program while the remainder divided equally between formal and informal programs.

The extent to which "Modern Programming Practices" were used on individual projects exceeded the amount reported by the companies as a whole. This probably reflects our request that projects employing these techniques be selected for the survey. The "newness" of these practices promoted us to include a reference in the table to assure a common understanding. Though every project reporting in this area employed some of the techniques (two small projects used only reviews) we did not really ascertain if a radical shift had taken place or whether many respondents were doing substantially what they've always done, but under a new name. HIPOs are, of course, a major departure, but, then again, their reporting uses were not very great. Table 11 reflects the results of our

Table 10

Quality Assurance Standards

Standard	Pct Company Wide	Pct Project Unique
Documentation	18	75
WBS code	44	25
Scheduling	38	31
Performance measurement	13	25
Requirements analysis	13	38
Preliminary design	19	56
Detail design	25	50
Coding	25	50
Test planning	13	56
Software verification	13	75
Reviews and audits	31	44
Configuration management	31	25
Discrepancy and correction	19	75
Software acceptance	13	56

Table 11

Modern Programming Practices

Practice	Pct Reported
Program manager authority [Black, 1977]	78
Reviews [Black, 1977]	81
Unit development folder [Ingrassia, 1976]	37
Design discipline and verification [Black, 1977]	48
Program modularity [Black, 1977]	70
Naming conventions [Black, 1977]	56
Structured form [Dijkstra, 1969]	15
Structured walk-throughs [Weinberg, 1971]	44
Structured analysis [Yourdon, 1975]	15
Structured design [Yourdon, 1975]	41
Chief programmer teams [Baker, 1972]	48
HIPOs [IBM, 1975]	11
Support library and facilities [Black, 1977]	44
Phase testing [Black, 1977]	81
Configuration management	78

survey. References cited contain the definitions for the purpose of this paper.

The software development tools/aids listed in Table 12 were used.

Table 12

Software Tools/Aids

Software Tools/Aids	Pct Used
Structured pre-compilers	7
Automatic flow charters	30
Library monitors	48
MACRO programming capabilities	44
On-line capabilities	63
Automatic test case generators	19
No software tools/aids employed	11

The test tools/techniques/methods listed in Table 13 were used [for definition of terms see Hartwick, 1977] (23 projects reported).

Table 13

Test Tool/Techniques

Tools/Techniques	Pct Used
Comparitors	22
Editors	39
Flow charting	74
Logic/equation generators	0
Program structure analyzers	26
Correctness proofs	9
Symbolic program executions	17
Initialization tests	52
Interaction tests	43
Arithmetic tests	70
Timing analysis	61
Branch logic tests	70

Organization.

There were a number of organizational variations and lines of authority used by the survey participants. Some form of project organization was the overwhelming choice with only one firm using the more traditional functional approach. A matrix organization in which the participants were detailed to the project manager for the duration of the task accounted for roughly 46% of the cases reported. Just who was detailed to whom usually depended on the orientation of the project manager. If the manager was from the data processing side of the house, the functional experts were detailed, and the reverse held true (data processing experts were detailed) if the manager was assigned to the functional activity. In two cases, however, a distinct entity was formed and both functional and data processing experts were detailed to form its complement.

A matrix organization in which work is accomplished through tasking line or staff agencies, rather than having individuals actually detailed, accounted for 24% of the projects reporting.

An often voiced concern of data processing (DP) managers engaged in development activity is the loss of control that may occur when programmers and analysts are detailed or assigned to functional managers. In 31% of the projects reported the DP manager did indeed lose control of DP people to some unspecified degree.

Table 14 (with 29 projects reporting) attempts to display this data. In constructing the array we separated the straight project organization from its matrix variant and subdivided the matrix form into its two basic orientations, task and detail.

Table 14

Organization Types

Organization Type (By Percent) Line of Authority	Function	Proj	Matrix Task	Matrix Detail
Line of Organization				
Under DP manager	0	24	7	21
Not under DP manager	3	3	11	18
Staff Organization				
Under DP manager	0	0	3	0
Not under DP manager	0	0	3	7
TOTALS	3	27	24	46

In every case, save one, the development organization was subdivided into teams, each under the direction of a technical leader (average 4 teams per project).

Though analyst programmer teams were by far the most common form, in some few

instances teams of straight analysts or straight programmers were also used.

Special purpose teams were also employed. Most important of these by percent of projects were: Test/product acceptance - 34%, integration - 28%, and interface - 14%.

Where test teams, or similar entities, were used, the head of this activity usually reported directly to the project manager. However, in 5 instances, the test team reported to a more senior individual, or to some other outside agency.

Staffing

Project Manager. Almost invariably the project managers were selected from in-house resources with approximately one-half coming from another software development project. As a general rule, appointment to the top project position was made by a senior manager not in the DP line of authority. The typical project manager had almost 11 years experience in data processing, though the range reported extended from 0 to 20 years. The years of experience in the functional area of the project averaged 7.5 years with a range of 0 to 16 years. Ninety percent of the project managers professed a working knowledge of FORTRAN and 76% some level of proficiency in an assembly language. Very few had any knowledge of either JOVIAL or COBOL.

The average age of project managers was 38, with a low of 34 and a high of 50. Only 19% admitted to ever having been a chief programmer. As Table 15 attests, they had an extremely high level of formal education. Degrees were primarily in engineering and mathematics. There were 3 physicists, 2 business majors, and no degrees in computer science.

Table 15

Project Manager Education Level

Degree or Degree Status	Pct
BS/BA degree	26
Masters degree	44
Masters degree plus 30 hours (or doctoral candidate)	15
PHD (or equivalent)	15

In most instances it was necessary to obtain some additional training, either prior to, or early in, the development cycle. Table 16 provides a breakdown by subject area and percent involved.

Table 16

Project Manager Training

Area	Pct Receiving Trng
Functional area of project	59
General data processing	50
Modern programming techniques	32
Project management	59
General management	59
A specific programming language	32
Other project related areas	9

The Software Development Staff. The source of programmers/analysts was: new hires from another company - 20%, new hires from school - 7%, in-house transfers from another project - 45%, in-house transfer from non-project oriented activity - 27%. The educational level of the programmers/analysts is given in Table 17.

Table 17

Programmers/Analysts Education Level

Degree or Degree Status	Pct
High school	< 1
AA degree or 2 years of college	11
Between 2 and 4 years of college	3
BS/BA degree	56
Masters degree	24
Masters degree plus 30 hours (or doctoral candidate)	2
PHD (or equivalent)	3

Training. In 78% of the projects reported the project manager was responsible for identifying training requirements for the development team. By far the most important source of training was on-the-job training; classes conducted by team members and hardware/software vendors came in a poor second and third. The training most often required was: Programming language - 47%, and operational system - 52%.

Of 8 projects reporting the use of a programmer support librarian, 5 reported filling the position with a clerk/programmer technician while the remaining 3

used programmers in this capacity.

The personnel turnover reported was: project managers - 64%, functional/user analysts - 43%, and data processing analysts - 14%.

Control

The process of controlling a software engineering project may well be the most talked about and least understood of all the project managers' functions. Many projects have failed because of the managers inability to adequately define precisely where the project was in the production cycle. Other development efforts have been cancelled, not for inadequate programming, nor lack of technology, but for the sheer frustration of all concerned in attempting to determine when, if ever, the project would be completed. This portion of the survey attempted to determine how project managers control and monitor the work from inception to completion.

Control Techniques. Early in the survey we asked project managers what type of tools they used in planning the project. In comparing the results of that question with the analogous query on control it was found that, by and large, the same tools were used in both functions. The only significant difference was the decrease in the use of workloading charts. (See Table 18)

Table 18

Systems Used In Project Control

System	Pct Reported
Milestone tracking	71
Work breakdown structure code	70
Workloading charts	36
GANTT charts	32
Modified PERT	29
PERT	4
Other	25
No systems used	21

Reporting. The three most frequently used manually prepared reports, the originators and recipients, and the percent of projects employing them were:

Weekly Activity:
Programmer/Analyst to Project Mgr - 38%
Project Mgr to Senior Management - 19%

Project Status:
Programmer/Analyst to Project Mgr - 19%
Project Mgr to Senior Management - 43%

Significant Change:
Programmer/Analyst to Project Mgr - 5%
Project Mgr to Senior Management - 14%

A number of managers reported using an automated system to monitor the development effort. In 54% of the projects this system was used to accumulate and display data such as manhours by activity. An activity was defined as a part of a task: flow diagramming, coding, etc. Thirty-one percent of the projects reported using a task oriented system. Beyond that, system software was used to check individual programs, the most common capabilities being listed in Table 19.

Table 19

Automatic Software Monitoring Capabilities

System Capability	Pct Reported
Count complies per module	5
Count lines of code produced	15
Check adherence to coding conventions	30
Check for use of standard data names	25

Other projects reported using productivity indexes, however, these were manually derived. The most frequently used indexes were: line of complete code per manhour, modules completed, and program errors.

Table 20 lists various stages into which a software development project may be divided. Our question asked: "Which of these were recognized as separate and distinct phases?". The percentages indicate that dividing projects into phases is one of the major methods the manager employs in "getting a handle" on the total effort.

Work Assignment. Tables 21 and 22 provide an encapsulized report on how work assignments from project manager to team chief to individual worker are generally handled. Table 21 indicates how task assignments were made to teams and individuals, whether completion dates were specified, and whether the task was defined in the context of the project. The questions that provided the framework for Table 22 attempted to ascertain the degree of interaction or feedback permitted in establishing required or estimated delivery dates.

In 63% of the projects reporting, tasks were assigned to teams and individuals as soon as they were sufficiently developed and defined. Twenty-four percent reported making work assignments as resources became

available and the remaining project managers made task assignments on a regular recurring basis, e.g., weekly, bimonthly, etc.

Table 20

Software Development Phases

Phase	Pct Reported
System definition	56
Requirements definition	81
System design	93
Module design	70
Coding	85
Module test	78
Sub-system integration	70
System integration	85
System test	93
Operation	63

Table 21

Method of Assigning Task

Action	Pct Frequency Reported Most of the Time	Pct Frequency Reported Seldom/Never
Task assignment given to the project team in writing	77	23
Required completion dates were included with each team assignment	92	8
Team assignments were prepared in such a way that the relationship of a task to the next higher level task was clearly delineated	62	38
Task assignments were given to the individual team members in writing	54	46
Required completion dates were included with each individual team member's assignment	79	21
Individual team member written task assignments were prepared in such a way that the relationship of a task to the next higher level task was clearly delineated.	67	33

Table 22

Determination of Completion Dates

Action	Pct Reported
Time and task assigned	33
Time and task assigned with verification or modification of time allotted being provided by individual team members as a matter of procedure.	37
Individual team member provided a time estimate for each task assigned	15
No effort was made to determine time requirements for individual tasks	19

Formal Review. In only 7% of the projects reporting were no formal reviews held. The average number of reviews held per project was between 3 and 4 with many projects holding 5. There were no surprises. As a general rule, the more costly the project, the more reviews. Table 23 shows the types of reviews and the percentage of projects that reported using them. An additional column has been added to show the percent of projects that used the review mechanism in establishing a baseline.

Table 23

Formal Reviews/Baselines

Reviews	Pct Reported Using	Pct Reported Establishing Baseline
Systems requirement review	56	18
Systems design review	63	9
Preliminary design review	85	32
Critical design review	78	32
Formal qualification review	30	0
Other reviews	7	0
No formal reviews conducted/ no baselines	7	27

Table 24 attempts to answer some basic questions about the review proceeding, the degree of formality and the attendance.

Informal Reviews. Reviews between the

Table 24

Formal Review Proceedings

Action	Pct Frequency Reported Most of the Time	Seldom/Never
Formal documentation was provided in advance of each review	88	12
Reviews took place on schedule	85	15
Top management attended formal reviews	42	58
Customer/user attended formal reviews	79	21
Project manager attended formal reviews	93	7
An independent review team was used	23	77

manager and his supervisor were also a common occurence. Table 25 shows that 96% of the project managers engaged in this activity with varying degrees of regularity.

Table 25

Frequency of Informal Reviews

Frequency	Pct Reporting
Daily	7
Weekly	37
Monthly	22
As required	27
No informal reviews	4

Configuration Management: Sixty-four percent of the development projects reported using some form of software configuration management system. However, about one-third of these were of an informal nature and only 42% resorted to a configuration control board. Baseline data was included in Table 23.

Walk-throughs. This technique or procedure as defined by Gerald Weinberg [1971] was used on a number of the projects. Table 26 reflects frequency, Table 27 scheduling, and Table 28 attendance.

Table 26

Walk-Through Frequency

Type of Walk-Through	Pct Reported
Design reviews	46
Coding reviews	38
Did not use walk-throughs	38

Table 27

Walk-Through Scheduling

Frequency	Pct Reported
Daily	4
Weekly	4
Monthly	0
As required	54

Table 28

Walk-Through Attendance

Attendance	Pct Type Walk-Through Design	Coding	Other
Peer programmers or analysts	45	45	18
Programmer or analyst trainee	14	14	5
Programmer or analyst supervisors	36	32	0
Project manager	37	18	0
Standard monitors	5	9	0
Top level manager	5	0	5
User/customer	9	1	0

In response to a question asking whether or not walk-through minutes were maintained it was found that records were kept in the design walk-throughs - 35% of the time, and in coding walk-throughs - 9% of the time.

Directing and Motivating.

Without exception the project manager was responsible for the technical quality of the system under development. Beyond that his responsibility and authority were not absolute. As can be seen in Table 29, the project managers' authority was most circumscribed when it came to hiring and firing, though this is not particularily surprising. We failed to determine if the project manager had certain prerogatives in selecting (in or out) present employees of the firm, but it is logical to assume that such authority did exist to some extent.

Table 29

Project Manager's Responsibilities

Project Manager Was Responsible For:	Pct
Technical quality	100
Hire and fire assigned personnel (within firm policy)	28
Evaluate performance of individual personnel	68
Administration, budget, etc.	80
Allocating computer resources	80
Meeting schedule commitments	96
Negotiating specification changes with customer	92
Making a profit (i.e., operating within budget)	48

The matrix organization, the most prevalent form in the projects reporting, undoubtedly contributed greatly to the next statistic (See Table 30) for in that form, the line of authority generally extends directly to the line or staff manager providing the resource.

Had we been aware of the extent to which matrix organizational forms were being employed we would have made some attempt to determine not only which, if any, "labeled" management or motivational techniques were being used, but, how their application differed when employed in a matrix, as opposed to a line or staff, organization.

At any rate, the results of the survey (See Table 31) indicated that the following techniques or procedures were being employed by project managers in the percentages indicated.

Approximately 10% of the projects surveyed reported at least some union membership among the project personnel, however, union membership had no recognizable affect on the projects.

Table 30

Project Manager's Authority

(A Composite Picture)

Position Description	Pct Assigned
Full time, report to project manager	23
Full time, report outside project manager's organization	43
Full time, outside contractor/consultant	24
Part time, report to project manager	4
Part time, report outside project manager's organization	5
Part time, outside contractor/consultant	1

Table 31

Management Techniques

Technique	Pct Used
Management by objectives [Drucker, 1954]	62
Management by exception	31
Job enrichment [Herzberg, 1977]	7
Incentive and suggestion programs	14
Open door policy	69

Deliverables and Successes.

The measure of success can be an arbitrary or subjective thing and so it was with our survey. There are of course degrees of success and one man's success may well be another man's failure. We asked, "Overall, how well do you think that this project met the project manager's major goals: to deliver on time, within budget, and meeting the requirement of the system, where the final software product is reliable, maintainable, and useable?"

We received the replies listed in Table 32.

Though every project was not yet complete they had all reached that state of "doneness" that the managers felt confident enough to provide an on-time or months-late input. Just under 50% of the projects

Table 32

Project Outcome

Goal Attainment	Pct
Extremely well	35
Very well	26
Good	13
Fair	4
Poor	4
Failed	13

Table 33

Percentage of Production Time

Area	Pct per Function
Requirement specification	11
Preliminary design	12
Detail design	21
Programming (and unit testing)	24
Integration	19
System testing	13

had, or were forecast to be, completed on time. A similar number were late and one project was delivered early.

The average number of months late was 8 and the average cost overrun was 25%. The major causes of project slippage in order of importance were given as: changing requirements, unreasonable or poor initial estimates, and limited authority over resources. Cost overruns were not limited to late projects, however, with 20% of the on time deliveries also exceeding initial estimates. The project that came in early also came in under cost. There is probably a moral there.

Since no measureable standards of reliability and maintainability had been established these requirements were met when the systems produced the desired outputs. Three projects provided a one year warranty and another indicated an implied warranty as long as the firm doing the development work was in the employ of the user. The average production rate measured in lines of code per programmer per day was 15. The cost per line varied from $8.00 to $200.00 - averaging $101.00. This number should not be construed as contradicting or refining the company figure given on a previous page nor does it convey anything more significant.

As a final question we asked: What percent of production time was spent in specific areas of the development cycle? The replies are summarized in Table 33.

Opinions on Improving Project Management

At the end of each section of the questionnaire we asked the participants for an opinion on what could or should be done to improve the state-of-the-art in software engineering project management. Our final section contains a selection of commentaries.

The proper preparation of requirement specifications was of major concern to most of the project managers. The answers given reflected the complaint that specifications are all too frequently incomplete, ambiguous, and inconsistent and plagued with seemingly arbitrary changes. In regard to this, one large manufacturer's comment was, "Make sure it [the requirement specification] is a joint commitment and effort by customers and developer". Another software development manager stated "If possible, the system definition should be firmed up earlier and held so that changes after the start of software development are only minor adjustments rather than redirection of the effort". No requirement specification was written in a requirement specification language. However, one participant believes that the form of the requirement specification is "...relatively unimportant, unless the customer wants to impose a design. Otherwise, any readable statement of the requirement will do, as long as it is clear and complete".

We received more comments on planning than on any other management function. Many of these comments were similar to this one: "Devote more effort to planning. Involve programmers/analysts to a greater degree. Require full documentation of plan. Continue to review plan as development proceeds". Another project managers approach was to: "Emphasize involvement and contribution of the "people doing the work" in the planning function".

When asked what changes should be made in the way technical decisions were arrived at concerning programming techniques, a dichotomy appeared to exist. At one extreme a need was expressed for "more automated tools; more formal walk-throughs"; and in supporting opinions other respondents suggested: "review what's in general use and publish recommended methods as a standard. No such document currently exists", and "get dollars, survey available methods, apply and weed out losers". However one project manager stated: "I would prevent outside 'experts' from imposing their pet techniques and controls in areas where they are inappropriate, misguided, and costly".

The matrix form of organization was clearly considered the best suited to software development efforts. However, by the opinion that the form of the organization was not as important as a "single clear leadership role" was a sentiment often expressed. Several managers, in response to a question on how they would organize for a software development in the future said "more and better planning".

Asked what action they would take if it was within their power to make changes or initiate research in the area of staffing, the project managers indicated they would: "experiment with more specialized staffing, i.e., with a number of functional roles rather than a single group of programmers/analysts", "investigate what personnel backgrounds lead or tend to lead to high performance ADP personnel" and implement studies in "how to maintain/improve communications".

Most of the project managers in responding on the subject of improved project control were in agreement with the contributor who stated: "establish definite packages of work and affect regular reviews with milestones".

On the subject of improvements in the way projects were directed, comments were very sparce. One analyst who provided us with an earlier quote firmly believes the necessity for "greater delegation of authority, responsibility", and "clearer assignment of responsibility".

Finally, the project managers were asked to list some of the lessons learned from their project. The responses to this question were voluminous and varied however, there was one response that clearly stands out as a summary: "establish requirements clearly before designing; establish design clearly before programming". "Give more attention to planning and to effort/cost estimates. Need greater documentation of decisions, specifications, [and] design".

ACKNOWLEDGEMENTS

The authors wish to acknowledge the following corporations, companies, and organizations for their cooperation in supporting this survey.

AFLC, Sacramento Air Logistics Center

AFSC, Aeronautical Systems Division

AUERBACH Associates, Inc.

Boeing Aerospace Company

Computer Science Corp, Defense Systems Division

COMSAT Laboratories

General Dynamics

General Electric Company

Grumman Aerospace Corporation

Grumman Data Systems Corporation

Hughes Aircraft Company

IBM Corp, Federal Systems Division

Jet Propulsion Laboratory

The Charles Stark Draper Laboratory, Inc.

The John Hopkins University, Applied Physics Lab

Lockheed Electronics Co., Inc., Data Products Division and Products System Division

Lockheed-Georgia Company

Lockheed Missiles and Space Company

McDonnell Douglas Astronautics Company

McDonnel Aircraft Company

NASA Headquarters

NASA Langley Research Center

PRC Information Sciences Company

Raytheon Company, Submarine Signal Division

RCA Corporation

Softool Corporation

System Development Corporation

Singer Company

TRW Inc., Defense & Space Systems Group

United Technologies Research Center

Vought Corporation

Acknowledgement is also due the following secretaries for their typing support and dedication of this paper:

Mrs Beryle E. McPheeters
Mrs Marianne Mueggenburg
Ms Lois Wauzinski

REFERENCES

Baker, F.T., "Chief Programmer Team Management of Production Programming", IBM System Journal, Vol II, Spring, pp 56-73 (1972)

Black, Rachel K.E., "BCS Software Production Data" BCS Report F30602-76-C-0174, (Prepared for Air Force Rome Air Development Center), Boeing Computer Services, Inc, Seattle (1977)

Dijkstra, E.W., Notes on Structure Programming (Aug 1969)

Drucker, Peter F., The Practice of Management, Harper and Row, New York (1954)

Hartwick, R. Dean, "Test Planning", AFIPS Conference Proceedings, 1977 National Computer Conference, AFIPS Press, Montvale, NY (1977)

Herzberg, Frederick I., "Orthodox Job Enrichment: A Common Sense Approach to People at Work", Defense Management Journal, (Apr 1977)

HIPO - A Design Aid and Document Technique, IBM Installation Manual, GC20-1851-1, IBM Corp (May 1975)

Ingrassia, F.S., "The Unit Development Folder (UDF) An Effective Management Tool for Software Development", TRW-SS-76-11, TRW, Inc (Oct 1976)

Perlis, A.J., Reported in NATO Conference Software Engineering Techniques (1969)

Weinberg, Gerald, The Psychology of Computer Programming, Van Nostrand Reinhold, New York (1971)

Yourdon, Edward, How to Manage Structured Programming, Yourdon, Inc, New York (1976)